Towards the Improvement of the Software Quality: An Enterprise 2.0 Architecture for Distributed Software Developments

Rafael Fernández[#1],Javier Soriano[#2],Xabier Larrucea[*3],Andrés Leonardo Martínez[†4],Jesus M. Gonzalez-Barahona[‡5]

[#]*Computer Networks & Web Technologies Lab., School of Computing, Universidad Politécnica de Madrid, 28660, Boadilla del Monte, Madrid, Spain*

[1]rfernandez@fi.upm.es

[2]jsoriano@fi.upm.es

[*]*European Software Institute (ESI), Bizkaia, Spain*

[3]xabier.larrucea@esi.es

[†]*Telefónica Research & Development, 28043, Madrid, Spain*

[4]jsoriano@fi.upm.es

[‡]*GSyC/LibreSoft, Universidad Rey Juan Carlos, Madrid, Spain*

[5]jgb@gsyc.es

***Abstract*— Software development is tightly dependent on the tools available for supporting its processes. Organizational and sociotechnical peculiarities such as indefinition of roles, geographically distributed development teams, new business models and diverse cultural interactions steer these tools. Software development supported by web-based services, built on top of Web 2.0 technologies, is emerging as a new paradigm for distributed software development. New generation software forges (web-based development environments) such as EzForge are becoming the infrastructure that provides the required features for hosting collections of software development projects. They are composed of an integrated set of tools, interacting in a mashup-like environment, each one suited for a specific task, and therefore simple enough to keep total complexity low. An adequate selection of tools helps developers to focus on the implementation of the requirements, while at the same time they cope with complex information coming from many individuals and organizations. The complexity of distributed software development requires a controlled and a strong collaboration amongst developers, which has to be supported by the selected architecture. Moreover, an increased demand on quality assurance is required by the many organizations aiming to achieve a certain quality level. A new architecture based on the Web 2.0 core ideas and methods overcomes these challenges in software development, representing a cornerstone to achieve satisfactory results in this ambitious environment.**

***Index Terms*— Web/Enterprise 2.0 Distributed Software Development, Process Definition, EzForge, Quality**

I. WEB/ENTERPRISE 2.0 TECHNOLOGIES AND QUALITY ASSURANCE

Nowadays many organizations are worried mainly about two issues: collaboration and quality assurance. As global market opportunities and competition increase, collaboration is becoming more and more essential for improving productivity and accelerating innovation at the personal, team, group, enterprise and business coalition levels. Many enterprise collaboration platforms have already been developed and successfully deployed in both large, and small- and medium-sized enterprises (SMEs). Enterprise collaboration has recently come to benefit from the emergence of an enterprise-oriented specialization of the Web 2.0 vision, commonly referred to as Enterprise 2.0 [1], providing new models and tools for emergent collaboration and co-creation. Enterprise collaboration is thus being enhanced by virtual communities that leverage social linking and tagging tools (such as those for social networking, social bookmarking and social search), user-contributed content management platforms (like enterprise Wikis, blogs and forums), tools to leverage user opinions (such as those supporting commenting and voting), subscription-based information distribution tools (such as corporate RSS feeds), etc. Used in the context of a carefully engineered collaboration strategy, these technologies provide a wealth of collaborative services for software developers [2].

On the other side, quality still represents a nightmare for too many organizations. In fact, quality cost is one of the most important considerations in software production [3], [4]. Quality assurance practices and software products quality represent in most of cases the forgotten requirement, and it is becoming a hard task to select an appropriate infrastructure capable of fulfilling, at the same time, customers' requirements and some level of quality assurance. The resulting solutions are usually defined in terms of which functionalities are exposed, and the question about what is the quality required and how do we achieve this quality are effaced from stakeholders' memory.

This situation is found in a broad range of scenarios, such as consultancy, in-house and outsourcing developments. The evolution of Internet technologies such as Web 2.0 and mashup platforms is supporting collaboration mechanisms, but at the same time they need to fulfil quality models requirements (e.g., Capability Maturity Model Integrated-

CMMI) [5]. In order to facilitate the fulfilment of these two challenges, we present in this paper a reference architecture that combines the evolution of these new Web/Enterprise 2.0 technologies, the mashup philosophy, and quality assurance facilities. The resulting collaborative environment is enriched with the savoir-faire (knowledge management) in software production environments.

The remainder of the paper is organized as follows. Section II presents an overview of EzForge, our Web 2.0-based networked forge. We then concisely set in section III the basis for enriching the forge with *savoir-faire* in software production environments. Section IV focuses on Method Engineering and the EzForge architecture from a holistic view, and describes the reference architecture for distributed quality software development. Section V elaborates on why this approach covers the main quality practices, and finally, we conclude the paper in section VI.

II. EzForge: New generation of Networked Forges Supporting Collaborative Software Development

Organizations tend to behave like dynamically reconfigurable networked structures that carry out their tasks by means of collaboration and teamwork. Effective teamwork is an essential part of any non-trivial engineering process, and collaborative capabilities are an essential support for these teams. Software development is no exception; being in itself a collaborative team effort with its own peculiarities. Both in the context of open source software development projects and in organizations that develop corporate products, more and more developers need to communicate and liaise with colleagues in geographically distant areas about the software product that they are conceiving, designing, building, testing, debugging, deploying or maintaining. In their work, these development teams face significant collaborative challenges caused by barriers raised by geographic distance, time factors, number of participants, business units or differences in organizational hierarchy or culture that inhibit and constrain the natural flow of communication and collaboration. To successfully overcome these barriers, they need tools to communicate with each other, and coordinate their work. These tools should also take into account the functional, organizational, temporal and spatial characteristics of this collaboration. Software product users are now becoming increasingly involved in this process, which means that they should also be considered. In response to this necessity, forges are gaining importance both in the open source context and the corporate environment.

Following the ideas in [6], a forge can be described as a kind of collaborative development environment (CDE) that provides a virtual space wherein all the stakeholders of a software development project, even if distributed by time or distance, may negotiate, brainstorm, discuss, share knowledge, and work together to carry out a software product and its supporting artefacts. It integrates multiple collaborative tools and resources, offering a set of services to aid all the stakeholders in the software development area, including managers, developers, users, commercial software manufacturers and software product support enterprises, to communicate, cooperate and liaise. Forges consider the social nature of software development and assure that the people who design, produce, maintain, commercialize and use software are aware of and communicate about the activities of the others simply, efficiently and effectively, also encouraging creativity and driving innovation. In doing so, forges provides with a safe a centralized solution conceived to optimize collaborative and distributed software development generally based on Internet Standards. This solution serves a number of essential purposes, including:

- Holistic integration of disparate collaborative processes and tools through a collaborative environment,
- Expansion of visibility and change control,
- Centralization and administration of resources, and
- Reinforcement of collaboration, creativity and innovation.

A. *EzForge*

The appearance of Enterprise 2.0-based forges, such as EzForge [7], supported by Enterprise Mashup platforms (EzWeb in this case, http://ezweb.morfeo-project.org/lng/en) and Gadget Development Environments (FAST in this case http://fast.morfeo-project.eu), enable software development teams to find, customize, combine, catalogue, share and finally use tools that exactly meet their individual demands. Supported by the EzForge platform, they can select and combine development tools hosted by third parties rather than buying a pre-determined, inflexible and potentially heavyweight software development environment.

EzForge, as the main part of the proposed architecture, is based on the idea of considering forges not as single sites providing a monolithic set of services to host projects as isolated silos of knowledge, but as a collection of distributed components providing services among which knowledge is shared. Each project decides on its own customized set of services, and users can configure their own mashup-based user interface. The main components are:

- Integrated systems. Those specifically built to feet into the proposed framework. Their interfaces will be described below.
- Legacy systems, which can be semantic or non-semantic. They are pre-existing systems that have to be integrated in the networked forge. The former offer some kind of semantically tagged information (RDF channels or some other XML markup) usually through REST interfaces. The latter provide just a non-semantically annotated HTML interface.
- Client components. Usually web browsers, interacting with the rest of the system via HTTP, will provide the user interfaces. They can as well be other systems capable of interfacing to RDF channels via HTTP, such as plugins for IDEs.
- Connectors. They connect non-semantic legacy components to the rest of the system. They usually parse HTML pages translating them into RDF channels,

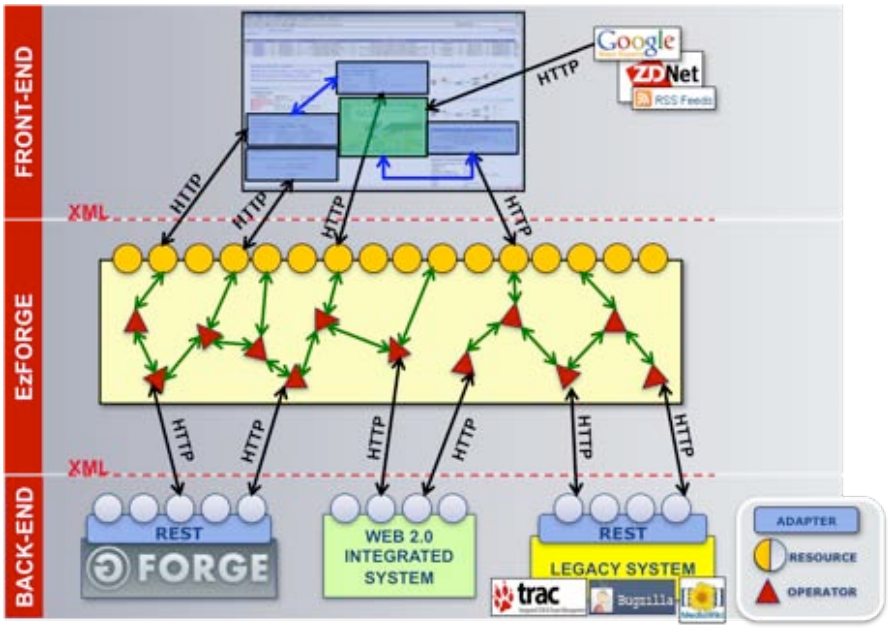

Fig. 1 EzForge Architecture

and receive REST invocations converting them into HTTP interaction with the legacy service.

- Adapters. They interface to semantic legacy components. Since those already provide XML access, adapters just translate and ``adapt'' the XML information of the legacy system to the RDF conventions used in the forge.
- Operators. They are made of aggregators, filters, and others. All of them are extended versions of the traditional ones, because they not only collect RDF channels, they also process them in several ways.
- Locators. Used as name services, allow for the registration of specific components, and offer their descriptions to clients and interconnecting components. Usually, each networked forge will maintain at least one locator, but each component can be registered in as many locators as needed.
- Catalogues. Components available for a certain community register in the corresponding catalogues. They are used by project administrators to select the set of components they will use by default, and by users to locate the components for their personalized forge. Depending on access policies, users can drag and drop information from a catalogue to a locator, or add new information to a catalogue.

Fig. 1 depicts the 3-tier EzForge architecture. The **back-end tier** is where integrated and legacy systems reside. Several connectors or adapters can work with the same legacy service, providing different interfaces to it. Conversely, a given connector or adapter can work with several instances of the same kind of legacy service, providing the same interface to several sites. It is important to realize that the EzForge architecture imposes no limitations to where the different components may be hosted. Those systems have their own set of basic forge services, such as source code management, wiki, and issue/bug tracking services, and they are integrated into the forge following a Web 2.0 approach consisting in the wrapping of their legacy services by a uniform layer of resources. These resources are components designed following the REST architectural style [8] that can be accessed through an URI via HTTP. Integrated systems already follow this approach, while legacy systems needs adapters to perform this task.

Thanks to the aforementioned layer of resources, the **EzForge tier** can access them to gather and process their data by means of special resources called operators, elements designed to get data from resources and use it to produce new data that can be processed by other resources, enabling their remix and composition. This way, EzForge creates the set of resources that will be delivered to end-users.

Once the EzForge tier has its forge resources set, final users are empowered by allowing them to design their own front-end (the **front-end tier** or forge user interface) by means of composing user interface building blocks called gadgets, which are endowing with the forge resources. Following this approach, users can mix and compose forge resources on their own, allowing them to choose the best resources to meet their needs. User can even include external resources, such as Google Maps or RSS feeds, into their UI, using all of them as a whole. They will use whichever resources they like to create ad hoc instant forge UI, encouraging resources mashup, and following the DIY (“do it yourself”) philosophy.

III. Savoir-faire in software production environments

More often than can be imagined, developers are plunged in an ocean of tools and procedures required in their daily work. Until this point, in this paper we have defined a forge as

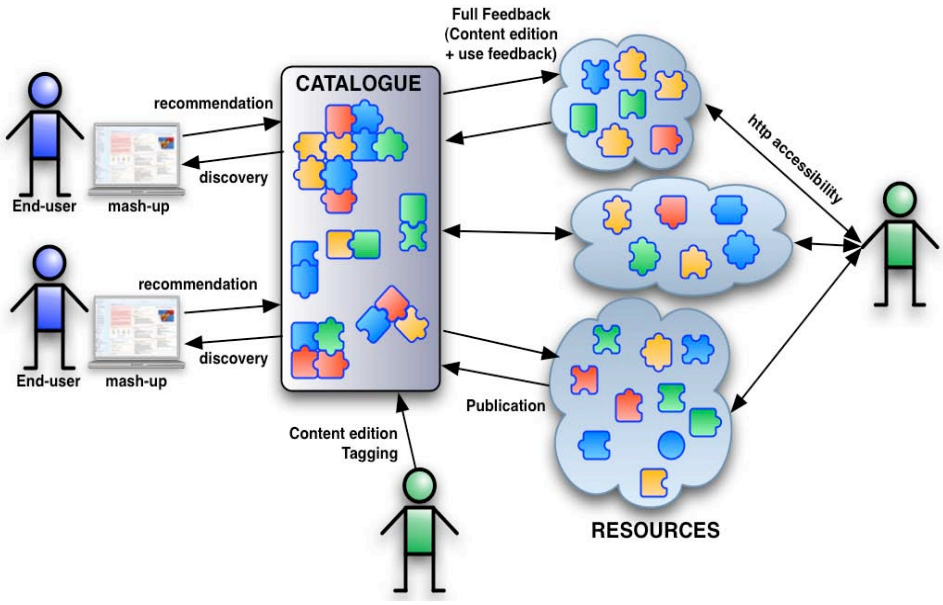

Fig. 2 Cataloguing Resources

a development platform, with project responsibilities delegated to developers, and without taking into account the management of their know-how. How do we materialize the know-how of your developments? How can we assure that our software products are developed as defined by the organization? These questions represent some of the factors that guide organizations to consider the materialization of their know-how and their internal procedures in some structure that assists them to avoid or overcome barriers and hurdles rising during their work. For example, some of these elements are the integration of new developers within development teams, and the quality assurance with respect to the requirements of quality models such as CMMI.

This is a cornerstone in our software development, and it is a part of the knowledge management (KM) broached by our architecture. One of the competitive advantages for organizations is their know-how, their human capital. Therefore we need to convert tacit knowledge into explicit knowledge, in order to share information and to promote the savoir-faire within the organizations. In the area of KM Peter M. Senge [9] defines “learning organizations”, and states five interrelated disciplines for the creation of smart and competitive organizations. In our approach we have used the method engineering approach as the way to make explicit tacit software production processes and methods in order to spread knowledge within the organization.

Method engineering [10] is used for several software developments and approaches [11], [12]. We have adopted Software Process Engineering Metamodel (SPEM) 2.0 [13] as a language for the definition of software development processes, and the Eclipse Process Framework (EPF) as a tool support for defining processes and methods in a Eclipse-based environment. The main idea is to define a methodology that relates method elements to the EzForge resources required for the software development. The huge number of resource-oriented services that are envisioned to be available in an Internet-scale networked forge will become unmanageable and thus useless for its users. Even if a repository service is provided, it will eventually become difficult for software development stakeholders to find out which resources (i.e. tool services) are appropriate for their development process.

This is the reason why we have created dedicated catalogues. In fact they provide navigation services for software development stakeholders and help them to find out which resources (i.e. tool services) they need to create the mash-ups they want. EzForge provides a user-contributed, “living” catalogue of resources founded on the Web 2.0 vision for user co-production and harnessing of collective intelligence (see Fig. 2). This would provide all stakeholders with a collaborative semantic Wiki, and tagging and searching-by-recommendation capabilities for editing, remixing and locating resources of their interest.

The catalogue sets out the knowledge available within a certain community for composing resources (e.g. a method from its fragments) in a graphical and intuitive fashion and for sharing them in a worldwide marketplace of forge services.

The catalogue allows users to create complex mash-up solutions by just looking for (or being recommended) “pre-cooked” or off-the-shelf resources and customizing these resources to suit their personal needs and/or the project requirements, interconnecting resources, and integrating the outcome in their development workspace. These decisions are defined during the development process definition.

Folksonomies of user-created tags will emerge and grow as users add information over time, acting as important facilitators of a useful marketplace of resources for the networked forge. Earlier approaches to service discovery and description like UDDI are not adequate to support human

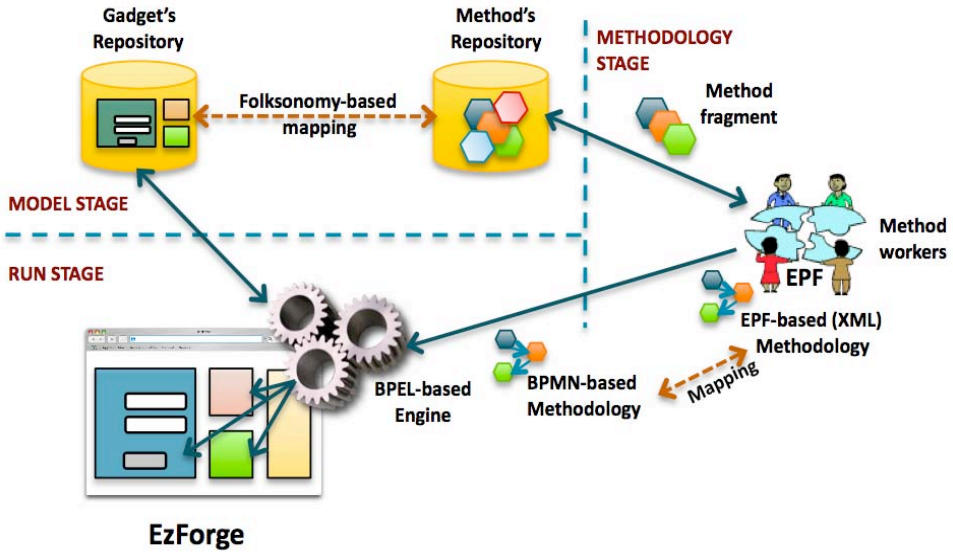

Fig. 3 A holistic view of Method Engineering and EzForge

beings in easy resource retrieval and evaluation. By contrast, the exploitation of collective intelligence and user-driven resource categorization is beneficial for users.

A straightforward application of our savoir-faire approach using the catalogue is split in four steps:

- **Evaluation of new developments**: taking into account previous experiences, method engineers evaluate a new software development. In this phase, Knowledge Management plays a relevant role identifying software development phases, tasks and problems that are resident in developers' minds. Method engineers should evaluate previous experiences and clearly specify what objectives of this new development are.
- **Selection of method fragments** based on previous experiences: method engineers select the appropriate set of method fragments fitting software requirements. In fact in this context each method fragment is related to a set of Web 2.0 resources. A basic catalogue contains the relationships between software processes and Web 2.0 resources. Each task is related to workproducts representing a resource and therefore method engineers could specify the appropriate tools support at each software development stage.
- **Composition of method fragments** in order to produce a software development process used in the organization. In this step the selected method fragments are composed defining a flow that it is guided by the methodology. This composition determines which are the selected resources at each stage of the development process. This approach is similar to Business Process Execution Language (BPEL) where web services are called following a specific order and sequence.
- **Deployment within the organization**. The resulting software development process is represented as a model and it is used by our forge. At this step, following CMMI terminology, the result represents a defined and managed Standard Software Process (SSP) for an organization. This is a requirement for organizations aiming to achieve compliancy with CMMI level 3.

This novel approach uses a method engineering approach in order to make explicit the savoir-faire in software developments within an organization. A catalogue contains relationships between method fragments represented by the methodology using SPEM 2.0, and resources that are represented within the forge as aggregators or connectors. Method engineers select the required method fragments needed for their software developments. In this context they select indirectly a set of aggregators and/or connectors that are related to specific resources. These resources are the basic tools within the development environment. Therefore we are reducing the gap between methodologies and software development tools support. This process allows the customisation of the resources and therefore the user's interfaces.

IV. Method Engineering and EzForge architecture: a holistic view

EzForge is a highly configurable and extensible user-centric collaborative software development tool that follows a novel mashup-based lightweight approach, provided by the EzWeb |14] core technology. Its user interface is defined by users theirselves, being able to make it up by assembling a set of small web applications called gadgets, which are the face of the services being offered by the forge. Up to now, there have been several attempts to bring mashup-based tools to the organizations [15], with satisfactory results. But with regard to software development, open source development tools don't take into account a key point in the software development within organizations: *quality*.

Method Engineering and EzForge architecture (Fig. 3) is compatible with the savoir-faire process defined previously and technically it is defined in three stages:

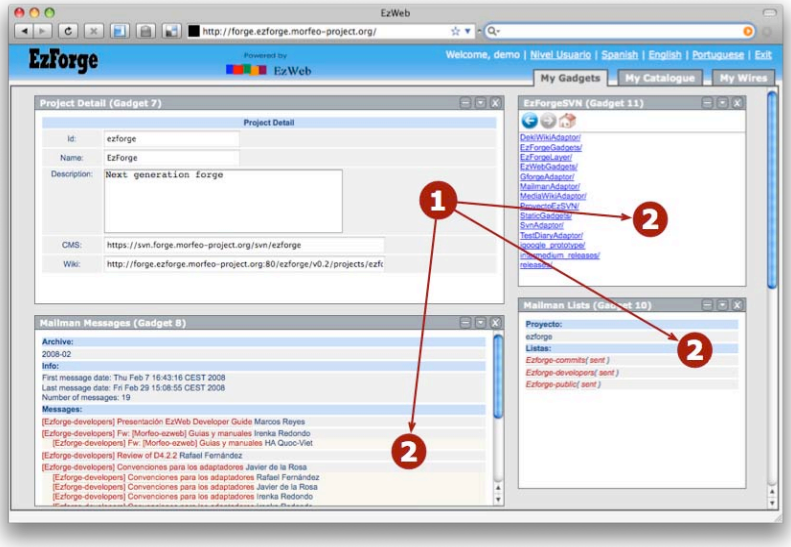

Fig. 4 Runtime execution overview

- **Model stage**. The goal of this stage is to link the available gadgets from the EzForge catalogue with the method fragments that exist in the method repository and that will conform lately the used methodology. This catalogue provides access in an automated way to EzForge catalogue in the execution level. For this purpose, we have developed a folksonomy-based mapping, which allows us to create that link by using social tagging techniques. By using these tags we will be able to choose the gadget or gadget group labelled with the method identifier in methodology run time in an easy way. Besides, it gives us a way to incorporate the organization internal knowledge about how things work better, as it is their own developers who carry out this tagging process.
- **Methodology stage**. It is in between model and run stages, and as we said before, it is where method workers select the method fragments that will make up the organization's methodology. To do so, method workers use the Eclipse Process Framework, which helps us to get, among other things, an XML representation of the methodology.
- **Run stage**. Once we've got the methodology, the next step is to put it in execution by means of a workflow engine. Thanks to this, EzForge can choose the appropriate and required development tools depending on the ongoing development phase.

Thus, our proposed reference architecture takes the advantages of method engineering and brings them all to EzForge, allowing companies and organizations to have a user-centric collaborative development tool which can guide its users through the development process.

An instance of the running application is shown in Fig. 4, using a web browser. Gadgets presented in this interface are those that have been defined by the method engineer when he was defining the organization's standard software process. Obviously there are some permanent gadgets in this interface, but most of them are configured during the model and methodology levels. Once we start/continue a software development, these gadgets are modified accordingly to a software development phase. In Fig. 4, the main gadget marked as “1”, acts upon the existence of gadgets marked as “2”. Moreover, there are some other relationships amongst gadgets, also shown in this figure. When an element in one gadget is selected, related elements in other gadgets are automatically selected. These relationships are not specified by the defined and managed methodology, they are implemented by the forge as a mean to achieve interoperability between gadgets. For this matter, EzForge follows the interoperability techniques proposed in [16].

V. Why this approach covers quality practices?

Managers are usually not worried about the technical architecture, but are more focused on costs and quality requirements used for the developments carried out in their organizations. Method engineering & EzForge architecture combine quality practices and a development infrastructure based on Web 2.0, ensuring quality aspects with low cost, open to new tools, representing an integrated environment, and overcoming new developer's barriers.

Why this approach covers quality practices? CMMI is one of most used quality reference model and it comprises two representations: staged and continuous. Whatever CMMI representation stakeholders select for their adoption, there is a common problem: a separation among process areas, due to a scarce tool support from a holistic perspective. Nowadays, engineering practices and process/project practices are separated, being one of the main tasks for adopters to assure that all process areas are coherent and consistent among them. Method engineering & EzForge architecture ensures quality

practices because the process defined and managed is used accordingly to its specification, and the development forge is guided by the methodology designed. In addition it also covers engineering and support process area because it provides an integrated development environment gathering requirements and configuration managements.

VI. On the road

The presented approach combines method engineering and Web 2.0 technologies in order to create a new generation of software developments tools and methods. Our approach starts from an explicit definition of the main tasks that a developer should carry out and its development environment is modified with respect to the organization's development process. This novel approach combines an extendable development environment based on Web 2.0 technologies with quality practices and tools. As results, we reduce the gap between development tools with low cost; we implement an extendable environment where we can select the appropriate tools support, and quality practices and tools are assured by this architecture. Web 2.0 technologies relevance is becoming during these last years a new wave in several environments and method engineering approach provides architecture to make explicit the knowledge managed by organizations. Our experience combining both approaches places us on the road for a new tool generation of software developments whose benefits are:

- **Facilitate collaboration in heterogeneous contexts**: Web 2.0 technologies facilitate software developments on the Web.
- **User interface configuration**: another advantage that comes directly from using method engineering and EzForge altogether is the possibility of generating the user interface based either on the methodology used. Thus, when creating a new project on the forge, it will be able to help the user to choose the tools to be used
- Help developers to **add new tools** within the development process.
- Compliancy at CMMI levels 2 and 3 through the definition of defined and managed standard software processes. The use of method engineering opens the door to the definition and management of the development processes. That is why EzForge will give support, for example, to the use of CMMI. This will make it possible to ensure that carried out developments will place the organization in a certain maturity level, allowing an improvement of the methodology used.
- Establish a relationship between process a project management tools (EPF) and engineering tools (forge).
- Provide a new tool in the knowledge management area through the use of method engineering.

Currently we are applying this approach in several test cases and projects. A step forward in this development is to provide facilities to the forge in order to collect all kind of metrics. This characteristic will provide a better understanding of our current developments.

Acknowledgment

This work is being supported in part by the Spanish Ministry of Industry, Tourism and Commerce under the National Research, Development and Innovation Program (VULCANO Project, Grant Agreement No. TSI-020301-2008-22, and EzWeb Project, Grant Agreement No. TSI-020301-2008-4), and by the European Commission under its 7th Framework Programme (FAST Project, Grant Agreement N0. FP7-ICT-2007-1-216048). The work has benefited from many discussions in the context of the Morfeo Open Source Community (http://morfeo-project.org). We thank all the people who contributed with ideas, comments, and criticisms.

References

[1] A. McAfee, "Enterprise 2.0: The dawn of emergent collaboration," *MIT Sloan Management Review*, vol. 47, no. 3, pp. 21–28, Spring 2006.

[2] J. Soriano, D. Lizcano, M. Canas, M. Reyes, and J. Hierro, "Fostering innovation in a mashup-oriented enterprise 2.0 collaboration environment,"*System and Information Sciences Notes*, vol. 1, no. 1, July 2007.

[3] L. Huang and B. Boehm, "How much software quality investment is enough: a value-based approach," *IEEE Software*, vol. 23, no. 5, pp.88–95, Sept.-Oct 2006.

[4] J. Campanella, "Principles of quality costs," *American Society for Quality Press*.

[5] S. S. Mary Beth Crisis, Mike Konrad, *CMMI Second Edition. Guidelines for Process Integration and Product Improvement*. Addison Wesley.

[6] G. Booch and A. W. Brown, "Collaborative development environments," in *Advances in Computers*, A. Press, Ed., vol. 59, 2003.

[7] (2008) The Ezforge project website. [Online]. Available: http://ezforge.morfeoproject.org/lng/en

[8] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, Irvine, 2000.

[9] P. M. Senge, *The fifth discipline*, Doubleday, Ed., 1990.

[10] S. Brinkkemper, "Method engineering: engineering of information systems development methods and tools," *Information and Software Technology*, vol. 38, no. 4, pp. 275–280, 1996.

[11] B. Henderson-Sellers, R.France, G.Georg, and R.Reddy, "A method engineering approach to developing aspect-oriented modelling processes based on the open process framework," *Information and Software Technology*, vol.49, no. 7, pp. 761-773, 2007.

[12] X. Larrucea, "Method engineering approach for interoperable systems development," *Journal Software Process: Improvement and Practice*, 2008.

[13] O. M. Group. (2007) Software process engineering metamodel (spem) 2.0 on OMG. [Online]. Available: http://www.omg.org/docs/ptc/07-11-01.pdf

[14] (2007) The Ezweb project website. an open source enterprise mashup platform. [Online]. Available: http://ezweb.morfeo-project.org/lng/en

[15] E. Driver, *Road map to an enterprise collaboration strategy*, F. Research, Ed., August 2 2004

[16] D. Lizcano, J. Soriano, R. Fernández, J. López, and M. Reyes, "Tackling interoperability in composite applications from an enterprise mash-up perspective," in *Proceedings of the 14th International Conference on Concurrent Enterprising (ICE 2008): 'A new wave of innovation in Collaborative Networks'*, T. Klaus-Dieter, P. Kulwant S., and R. Gonsalves, Eds. Centre of Concurrent Enterprise, Nottingham University Business School, University of Nottingham, UK., June 2008.